REMARKS

Claims 1-27 are currently pending in the patent application. The Examiner has objected to the Abstract of the disclosure. Applicant has reviewed the Abstract of the disclosure and believes that the language found therein reflects the claim language by expressly reciting the resetting at least one class from the first application, followed by having the second application initialize classes which have been reset from the first application prior to using those classes. Applicant believes that the Abstract of the disclosure is sufficiently detailed to assist a reader in deciding whether there is a need for consulting the full patent text for details. Accordingly, Applicant is not submitting an amended Abstract at this time. Applicant requests that the Examiner reconsider the objection and propose specific language if the Abstract is found inadequate upon review.

The Examiner has rejected Claims 1-27 under 35 USC 103 as being unpatentable over Holiday in view of Konuru. Applicant respectfully asserts that the combination of references does not teach or suggest the invention as claimed.

The present invention provides a system, method, and program product for allowing a virtual machine to run an application repeatedly without the need to reload and initialize all classes required for that application and/or to run a related application, that shares classes with a previously-run application, without having to reload and initialize all classes. The system, method, and program product, as taught and claimed, starts a first application including loading and initializing a set of classes for the first application; running the first application; resetting at least one class from the first application after the first application has finished running; and then allowing the second application to initialize the at least one class that has been reset, without having to reload the class or classes into the virtual machine. Applicant has amended the language of the independent claims to highlight the distinction that the second application may initialize the at least one class that had been reset without having to again load the at least one class into the virtual machine.

The Holiday patent is directed to a method and system for loading software for a Java application program into a virtual machine wherein the software is organized into packages and the packages are loaded in a determined order, wherein the order is determined to insure that no package is loaded before the package or packages upon which it depends. The Examiner has cited the process flow illustrated in Figure 4, and specifically reference numerals 416, 418, and 420, against the claim feature of starting a second application. Applicant respectfully disagrees. The Holiday method is to load one application by loading the packages and classes needed for that one application in a determined order (see: Col. 1, lines 6-8; Col. 3, lines 25-29, Col. 7, line 14 through Col. 8, line 51, which all refer to loading a single application) and the Figure 4 process flow is the process for loading the one application. While the process flow includes steps for checking for and loading additional class files (see: 418 and 418) and/or additional package files (see: 422), those class files and packages are part of the one application or are necessary for running the one application. There is nothing in Holiday which either teaches or suggests that a second application is being loaded. Since Holiday explicitly teaches loading packages

in order for only one application, Applicant respectfully concludes that Holiday does not obviate the claim feature of starting a second application on the virtual machine. Moreover, the claim feature expressly recites "said second application initializing said at least one class that has been reset from the first application without having to load the at least one class."

Applicant asserts that Holiday does not teach or suggest starting a second application and does not teach or suggest that a second application initializes at least one class without loading the at least one class. Holiday does teach that "image" information be stored with application packages and classes are loaded (see: Col. 9, lines 4-12). However, that "image" information comprises copies of the packages and classes of the loaded application "written out to a non volatile medium such as a hard disk file". In the event of a failure, the image information could be read out and the JVM could be restarted. Such failure recovery is, in fact, reloading the information into the JVM. Holiday expresses states at Col. 9, lines 15-16 that the JVM would need to read in the data and reinitialize each class. Clearly, therefore, Holiday is not teaching or suggesting a

second application initializing classes which have been reset without having to reload the classes.

The Examiner has acknowledged that Holiday does not teach or suggest the step or means for resetting at least one class from the set of one or more classes which were loaded and initialized for the first application after the first application has finished running. The Examiner has cited the Konuru patent against that claim feature. What the Konuru patent teaches is monitoring events and applying sequencing rules to ensure consistent reporting of events. The Examiner cites the Konuru teaching that class static fields are initialized and the teaching that once objects are allocated in memory, their raw memory address remains the same all through program execution.

Applicant respectfully asserts that the Konuru teachings do not teach or suggest loading classes for a first application, resetting the classes after the first application has finished running, and then reinitializing the classes by a second application. When Konuru teaches that class static fields are initialized, that does not mean that the classes are permanently stored. Moreover, when Konuru teaches that raw memory addresses for objects remain unchanged "all through program execution", that does not

teach or suggest that class objects are permanently stored. Rather, Konuru, like Holiday, loads classes for use by one application/program. Once that program completes execution, there is nothing in either Konuru or Holiday which teaches, or even suggests, that the classes be reset for use by a successive application.

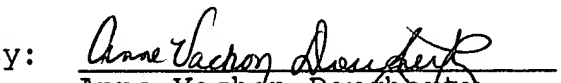
Applicant contends that, even if one were to modify Holiday with Konuru, one would not arrive at the invention as claimed, since neither patent teaches or suggests resetting at least one class, thereby allowing a second application to initialize the class without reloading it. If one were to modify Holiday with Konuru, the result would be a Holiday system wherein packages and classes are loaded in order for a first single application, wherein the events during program execution by that first application would be monitored and reported in order, and wherein the steps would be repeated for a second application (i.e., load everything in order and then monitor and report events in order). One would not, however, arrive at a system, method, or program product wherein classes for a first application are reset so that a second application can initialize those classes without the need to again load them. Applicant concludes,

therefore, that the combination would not obviate the invention as claimed.

Based on the foregoing amendments and remarks, Applicant respectfully requests entry of the amendments, reconsideration of the claim language in light of the remarks, withdrawal of the rejections, and allowance of the claims.

Respectfully submitted,

Alan M. Webb

By: 

Anne Vachon Dougherty
Registration No. 30,374
Tel. (914) 962-5910